LU02.A02 - Klassendiagramm "Bottle" umsetzen

Implementieren Sie das Klassendiagramm als Sourcecode.

Auftrag

1. Implementieren Sie das Klassendiagramm `Bottle` als Python-Code.
   Orientieren Sie sich für die Umsetzung am Beispiel "Door".
2. Halten Sie die BZZ Codingstandards für Python ein.
   Die Aufgabe ist nur dann vollständig gelöst, wenn Sie auch die BZZ-Vorgaben bezüglich «guter Programmierung» einhalten.

Vorgehen

Beim Lösen der Programmieraufgaben orientieren wir uns am TDD (test driven development). Die nötigen Testfälle werden durch die Lehrpersonen in Ihrem GitHub Repository bereitgestellt.

Sie finden weiter unten die genauen Anleitung für das jeweilige Vorgehen. Dieses ist bei der Implementation in allen kommenden Übungen anzuwenden.
Es ist wichtig, dass Sie sich dieses Vorgehen aneignen, da auch die Prüfungen so umgesetzt sind und die erreichte Note durch die Anzahl korrekter Tests gegeben ist.

Dauer

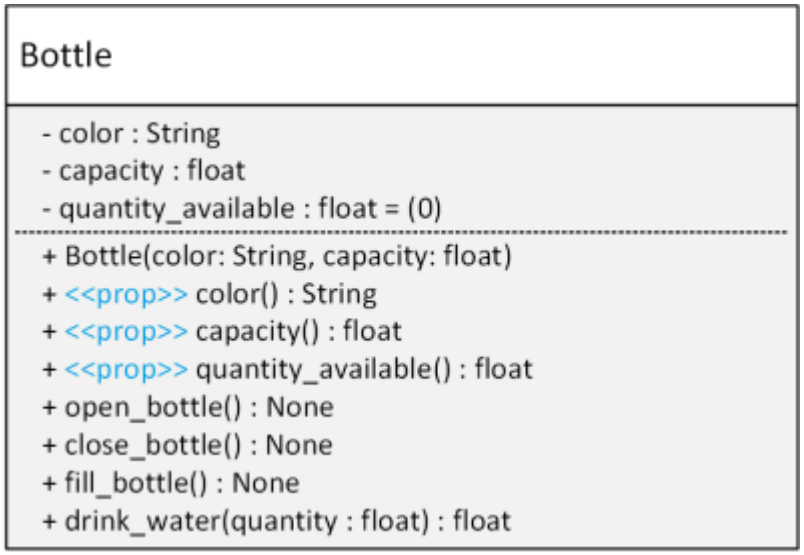
40 Minuten + Hausaufgabenzeit

Abgabe

Commit und Push auf github

Anleitung

Klassendiagramm



Beschreibung der Klasse

Die Klasse Bottle stellt eine einfache Trinkflasche mit 3 Attributen sowie einigen Methoden dar.

Attribute

- `color` ist die Farbe der Flasche.
- `capacity` ist die maximale Menge, die in der Flasche Platz findet.
- `quantity_available` hält die aktuelle Menge Wasser in der Flasche fest. Defaultwert = 0

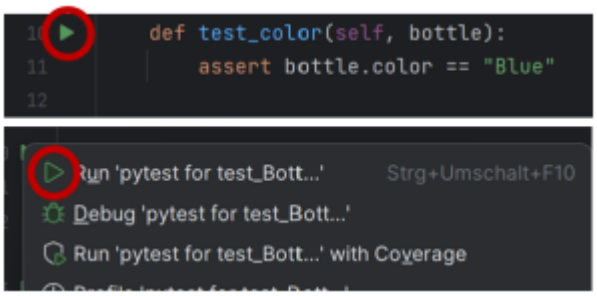
Methoden

- `__init__(…)` (der Konstruktor der Klasse Bottle) initialisiert die verfügbare Menge (quantitiy_avaible) auf 0, während Kapazität (capacity) und Farbe (color) der Flasche durch Parameter festgelegt werden.
- `color()` liefert die Farbe der Flasche. Wird als `@property` realisiert.
- `capacity()` liefert das Volumen der Flasche. Wird als `@property` realisiert.
- `read_quantitiy_available()` liefert die in der Flasche vorhandene Menge. Wird als `@property` realisiert.
- `open_bottle()` wird leer implementiert (Keyword pass verwenden)
- `close_bottle()` wird leer implementiert.
- `fill_bottle()` füllt die Flasche bis zum maximalen Fassungsvermögen.
- `drink_water(amount)` liefert die angeforderte Menge aus der Flasche, falls diese Menge noch verfügbar ist. Wenn die verfügbare Menge (quantitiy_available) kleiner ist, wir diese Menge geliefert und die Flasche ist leer.

Vorgehen

1. Akzeptieren Sie das GitHub Classroom Assignment.

2. Klonen Sie ihr Repository in die Entwicklungsumgebung.
3. Implementieren Sie den Konstruktor (`__init__(…)`) und initialisieren Sie die Attribute gemäss Beschreibung.
4. Implementieren und testen Sie das Attribut `color`
    1. Schreiben Sie die `@property` für `color`.
    2. Führen Sie in `test_Bottle_class.py` die Testmethode `test_color` aus. Sie muss fehlerfrei ablaufen.



    3. Das Ergebnis des Tests muss wie folgt aussehen:

```
============================= test session starts ==============================
collecting ... collected 1 item

test_Bottle_class.py::TestBottle::test_color <- test_bottle_class.py PASSED [100%]

============================== 1 passed in 0.01s ===============================
```

    Wird ein Fehler signalisiert, muss die Methode solange bearbeitet werden, bis der Test «passed» ist.
    Beispiel eines fehlerhaften Testlaufs:

```
============================= test session starts ==============================
collecting ... collected 1 item

test_Bottle_class.py::TestBottle::test_color <- test_bottle_class.py

============================== 1 failed in 0.12s ===============================
FAILED [100%]
test_bottle_class.py:9 (TestBottle.test_color)
'green' != 'Blue'

Expected :'Blue'
Actual   :'green'
<Click to see difference>
```

5. Führen Sie in Ihrer Entwicklungsumgebung einen **Commit** und einen **Push** aus.
   Wichtig: Immer dann, wenn Sie einen Test erfolgreich ausgeführt haben, pushen Sie das Programm. So ist sichergestellt, dass auf GitHub jederzeit lauffähiger (und bewertbarer) Code liegt!
6. Implementieren Sie nun Schritt für Schritt die weiteren Methoden nach dem gleichen Vorgehen.
    1. `capacity()` ⇒ `test_capacity`
    2. `read_quantitiy_available()` ⇒ `test_initial_quantity`
    3. `open_bottle()` und `close_bottle()` werden mit der `pass`-Anweisung (leere Methoden ohne Funktion) realisiert. Testmethode: `test_open_and_close_bottle`.
    4. `fill_bottle()` ⇒ `test_fill_bottle`. Überlegen Sie sich, wie Sie sicherstellen können, dass die Flasche ganz gefüllt ist.
    5. In der Methode `drink_water(…)` stellen Sie sicher, dass die gelieferte Menge korrekt ist. Dazu müssen Sie überprüfen, ob die angeforderte Menge (`amount`) in der Flasche verfügbar ist. Weiter müssen Sie sicherstellen, dass der Inhalt der Flasche um den Betrag verringert wird. Reicht der Inhalt nicht, wird einfach der Rest in der Flasche geliefert (und die Flasche ist leer)

1. Testmethoden: `test_drink_water_available`, `test_drink_water_not_avaible`, `test_drink_water_partial_avaible`

Zum Schluss führen noch einmal alle Tests aus. Geben Sie dazu im Terminal (Eingabeaufforderung) den Befehl `pytest` ein.

Das Testergebnis muss nun wie folgt aussehen:

```
============================= test session starts ==============================
collecting ... collected 8 items

test_bottle_class.py::TestBottle::test_color PASSED                      [ 12%]
test_bottle_class.py::TestBottle::test_capacity PASSED                   [ 25%]
test_bottle_class.py::TestBottle::test_initial_quantity PASSED           [ 37%]
test_bottle_class.py::TestBottle::test_fill_bottle PASSED                [ 50%]
test_bottle_class.py::TestBottle::test_get_liquid_available PASSED       [ 62%]
test_bottle_class.py::TestBottle::test_get_liquid_not_available PASSED   [ 75%]
test_bottle_class.py::TestBottle::test_get_liquid_partial_available PASSED [ 87%]
test_bottle_class.py::TestBottle::test_open_and_close_bottle PASSED      [100%]

============================== 8 passed in 0.02s ===============================
```

⇒ *GitHub Repo für externe Besucher*

GitHub Repository https://github.com/templates-python/m320-lu02-a02-bottle

Lernende am BZZ müssen den Link zum GitHub Classroom Assignment verwenden

René Probst, bearbeitet durch Marcel Suter

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m320_2024/learningunits/lu02/aufgaben/bottle

Last update: **2026/08/30 14:06**